LECTURE – 3

LAB ASSIGNMENT #1

1. Use the random number generator to generate a sequence of N random integers, where N can have value 100, 1000, 2000, 5000 ... 100000 ... *etc*.

2. Program the insertion sort algorithm to sort the above N integers. Input and output are both in the form of disk files. Test the algorithm for small N.

3. Measure the running time of the algorithm, and plot it as a function of N. Verify that the running time grows as N^2.

4. Repeat steps 2 and 3 with the merge-sort algorithm. Plot and verify that the running time grows as N*log(N). Compare the two plots. Do they cross?

ORDER NOTATION

We need a mathematical technique to understand and analyze the running time behaviour of algorithms; that is, the "time complexity" of algorithms. This is both a theoretical and practical necessity.

[Sometimes the concept of "space complexity" is also studied, but in this course we shall not get into that area.]

What is the meaning of "asymptotic efficiency" of algorithms?

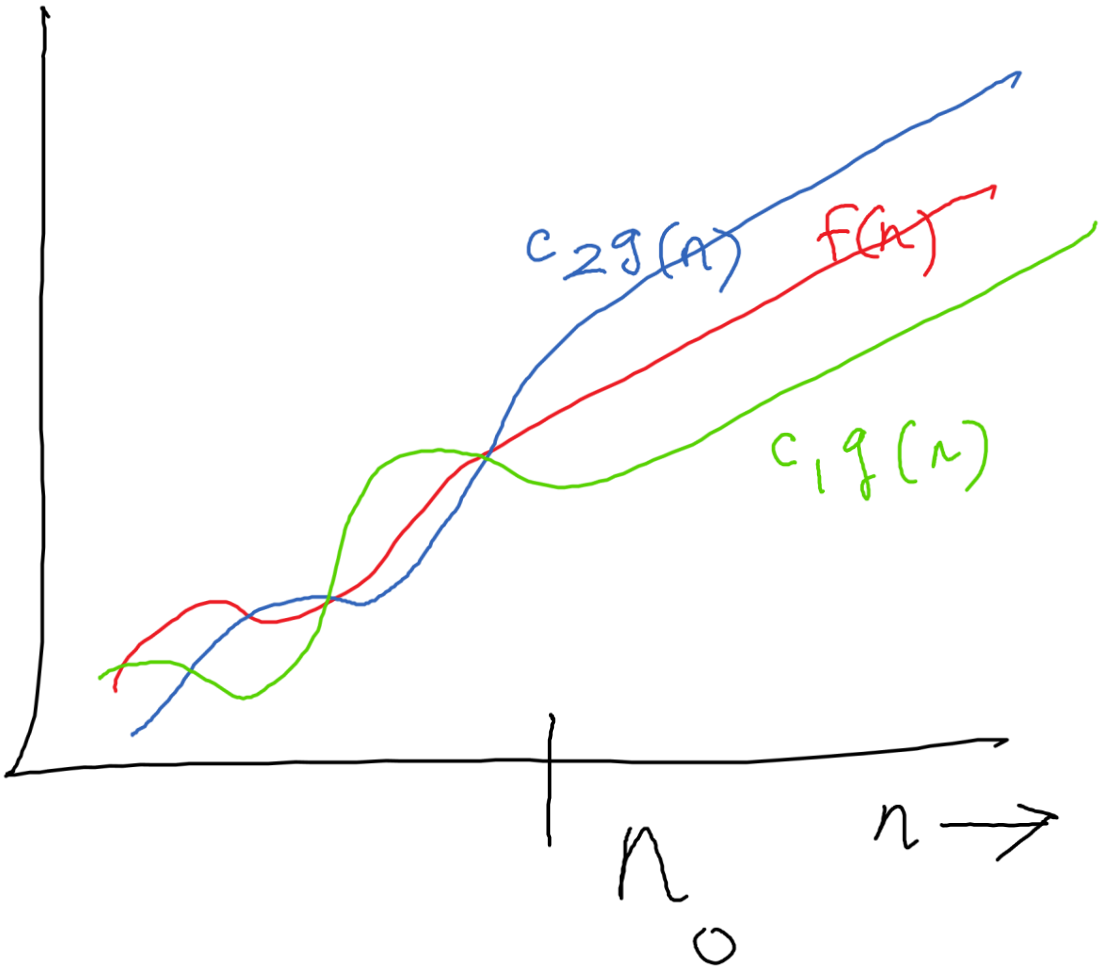
Recall the behaviour of a function such as $f(x) = e^{-x}$. This function goes to zero asymptotically; that is, ONLY in the limit as *x* goes to infinity.

$\Theta(g(n)) = \{ f(n) \mid$ there exist positive constants c_1 , c_2 and n_0 such that,

for all $n > n_0$, $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ $\}$

Technically, we must say that a function f(n) satisfying the above condition is an element of the set $\Theta(g(n))$, belongs to the set $\Theta(g(n))$, or is in $\Theta(g(n))$. But we often say, as a convenient shorthand, that $f(n) = \Theta(g(n))$.

Graphically, we can see that, for all $n > n_0$, the function f(n) is "sandwiched" between the two functions $c_1 g(n)$ and $c_2 g(n)$.



Order notation does NOT apply to “small" values of n, that is, for $n \leq n_0$.

Here g(n) and all possible functions f(n) are asymptotically non-negative. Recall that the running time of an algorithm can never be negative.

Examples:

(1) Prove that $n^2/2 - 3n = \Theta(n^2)$. See [CLRS].

It is shown that $c_1 = 1/14$, $c_2 = ½$ and $n_0 = 7$ suffice to prove the statement, but note that these values are not uniquely determined.

(2) How do we show that $n^2/2 - 3n$ **is not in** $\Theta(n^3)$?

(3) How about $n^2/2 + 3n$?

The notation $\Theta(g(n))$ can be applied to the best case, worst case and average case running times of an algorithm. The approach helps in choosing the most efficient algorithm from several candidate algorithms. But if an algorithm is to be run for only “small" values of *n*, this may not be the main criterion.